Deep TEN: Texture Encoding Network

Hang Zhang, Jia Xue, Kristin Dana

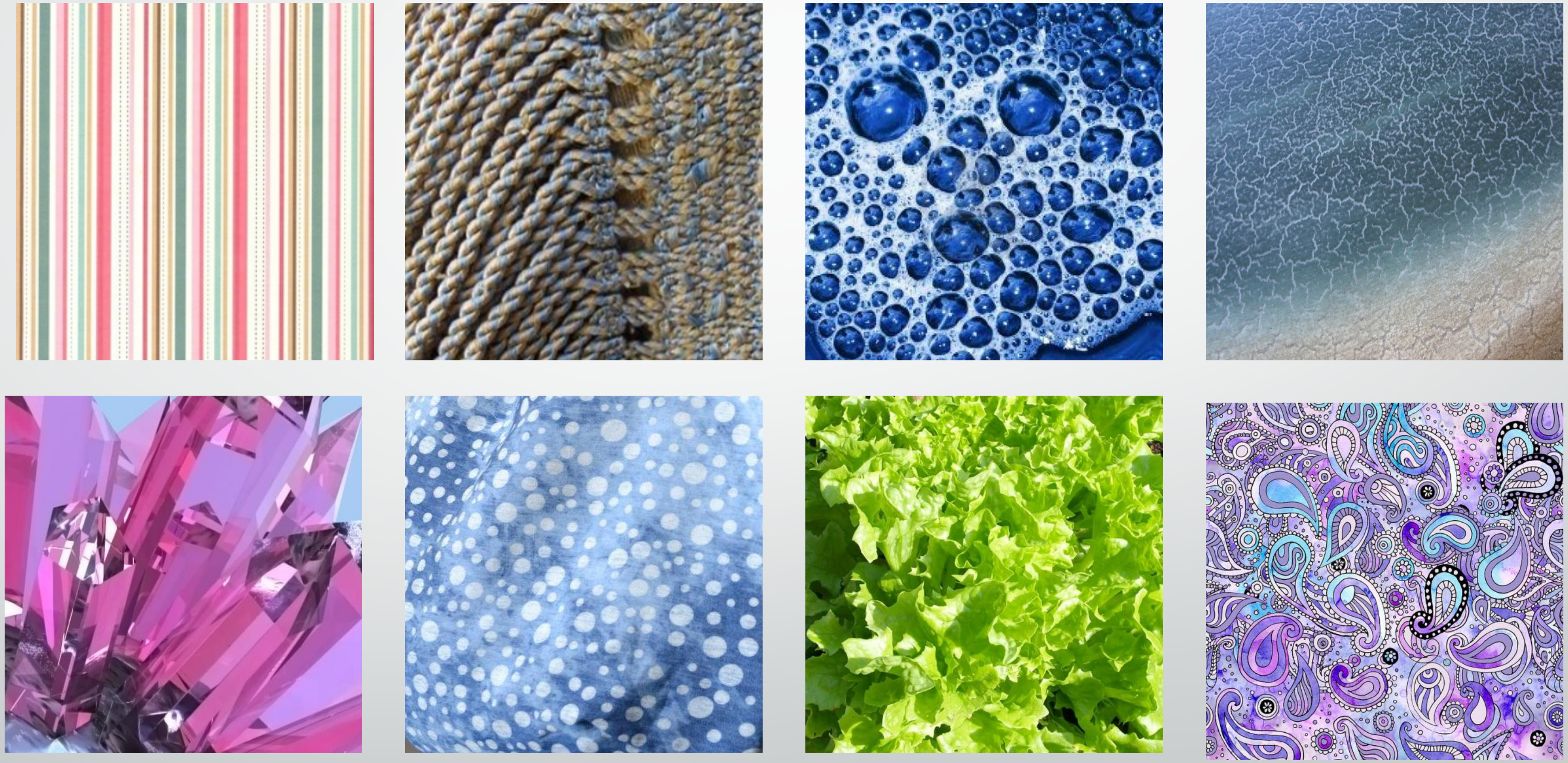
RUTGERS

Highlight and Overview

- Introduced Encoding-Net
  a new architecture of CNNs
- Achieved **state-of-the-art** results on texture recognition
  MINC-2500, FMD ,GTOS, KTH, 4D-Light
- Released the ArXiv paper (CVPR 17) and Torch Implementation (GPU backend)

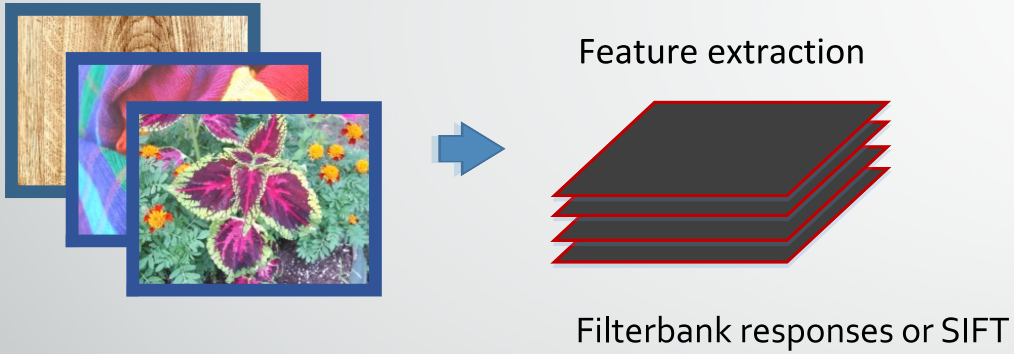
Challenges for Texture Recognition

- Orderless
- Distributions

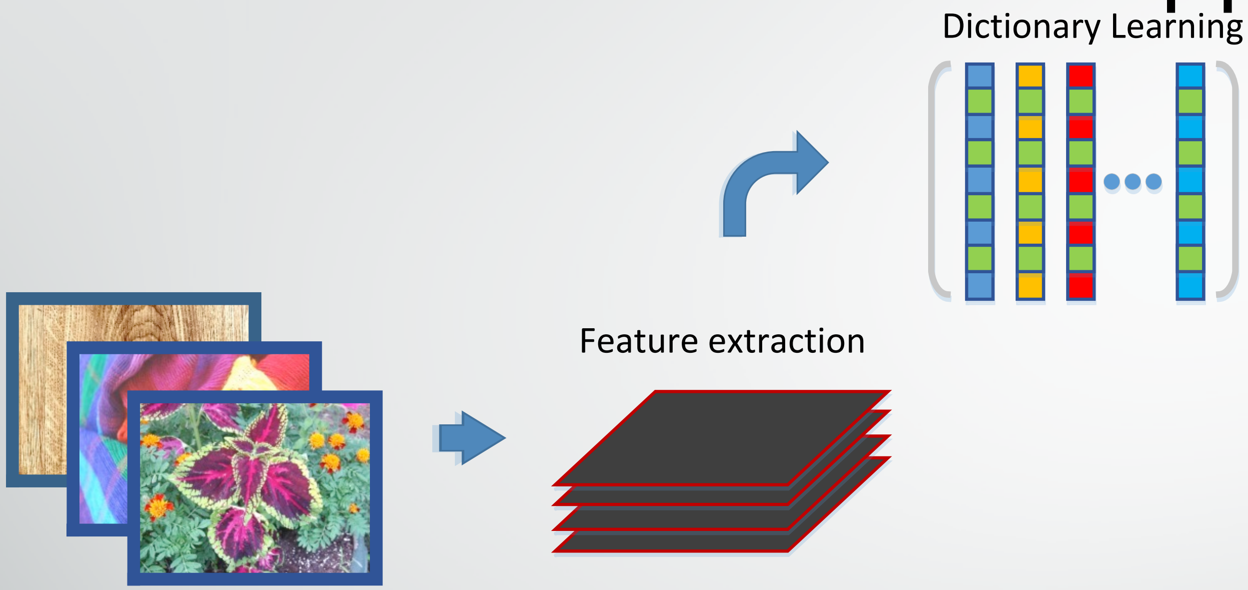


Classic Vision Approaches

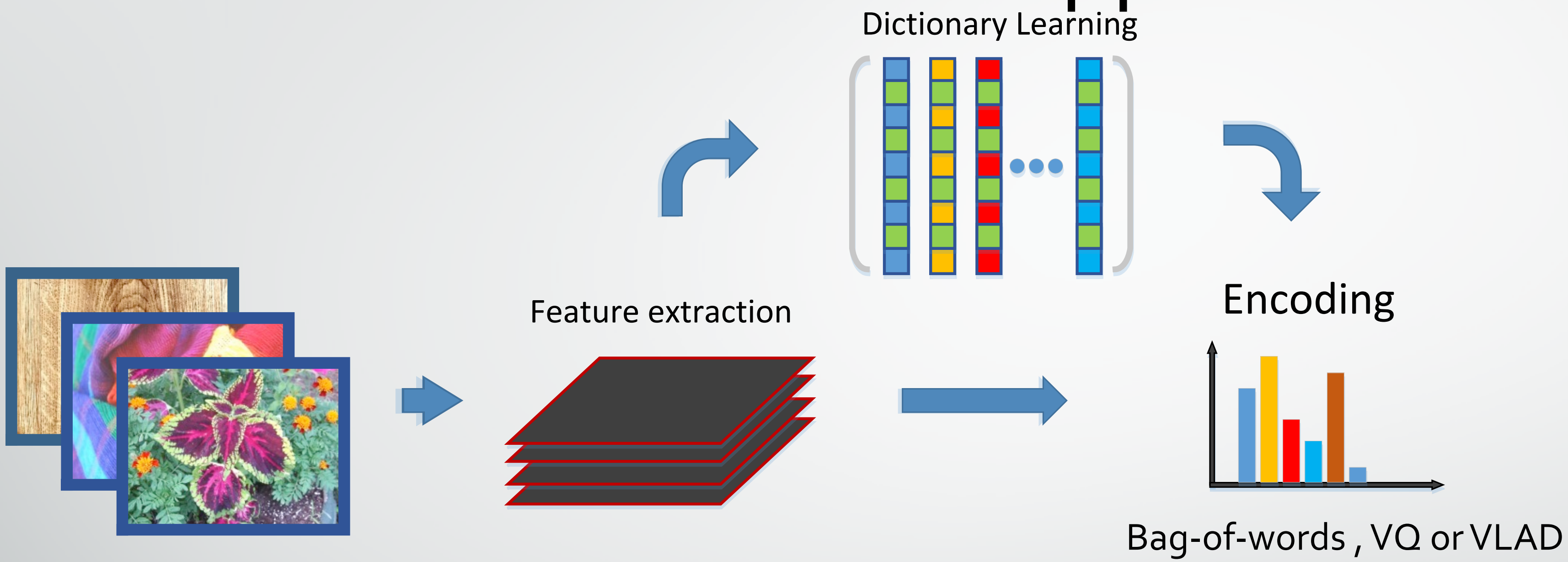
Classic Vision Approaches



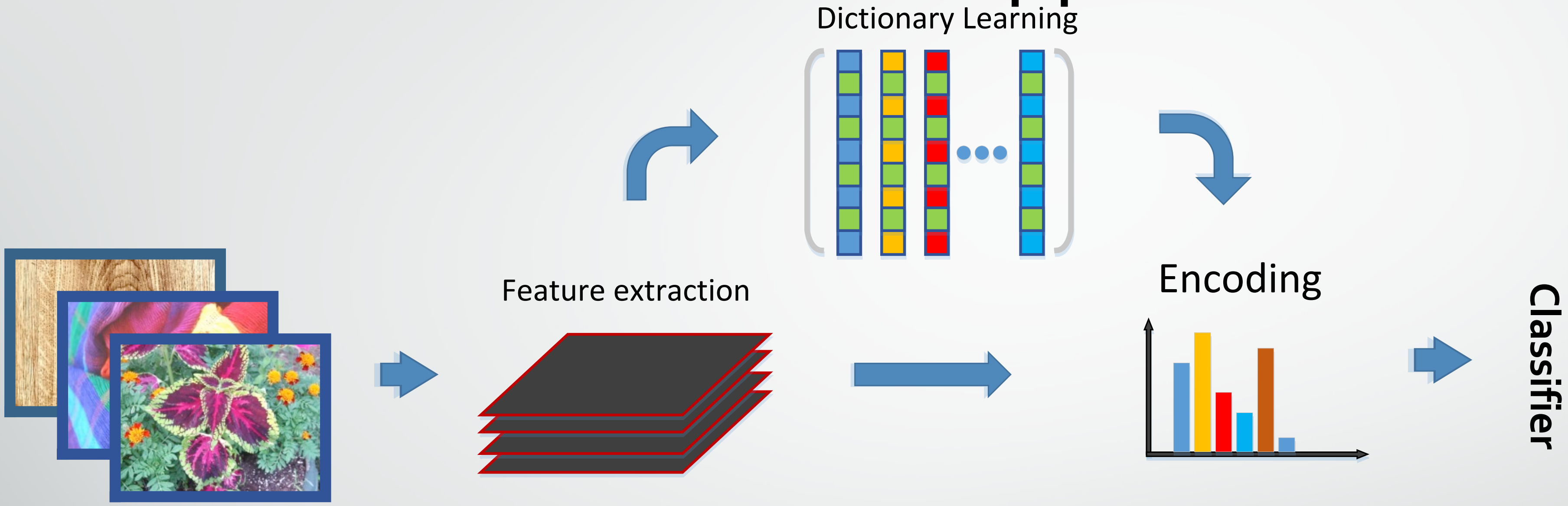
Classic Vision Approaches



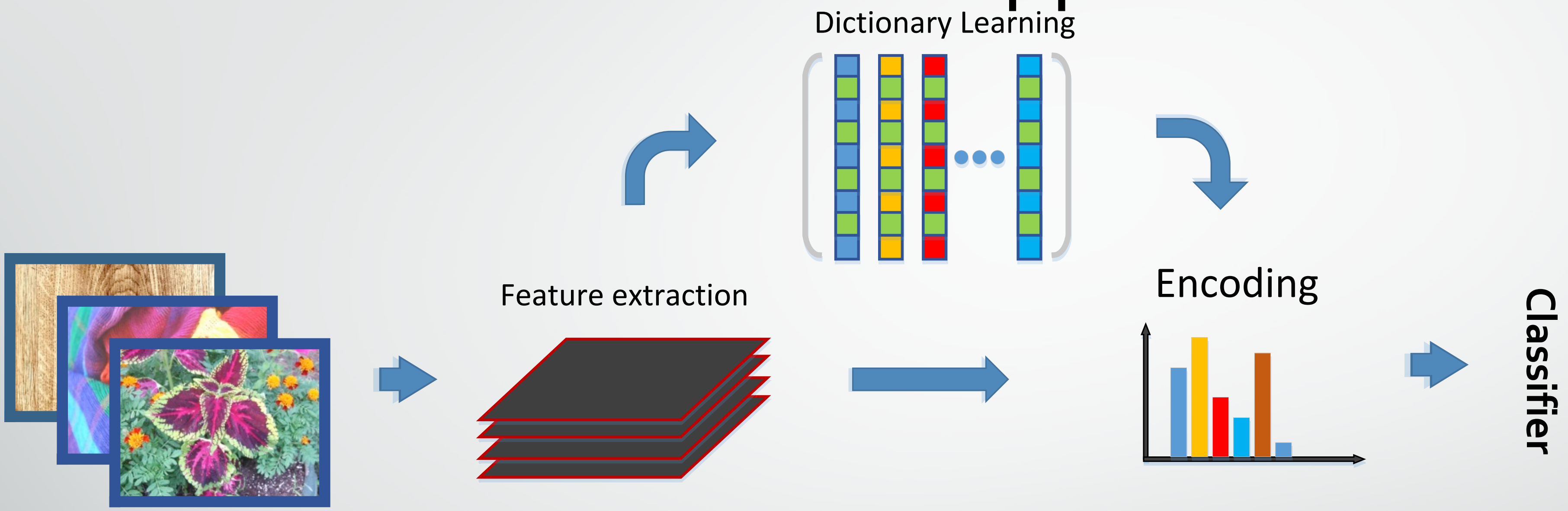
Classic Vision Approaches



Classic Vision Approaches

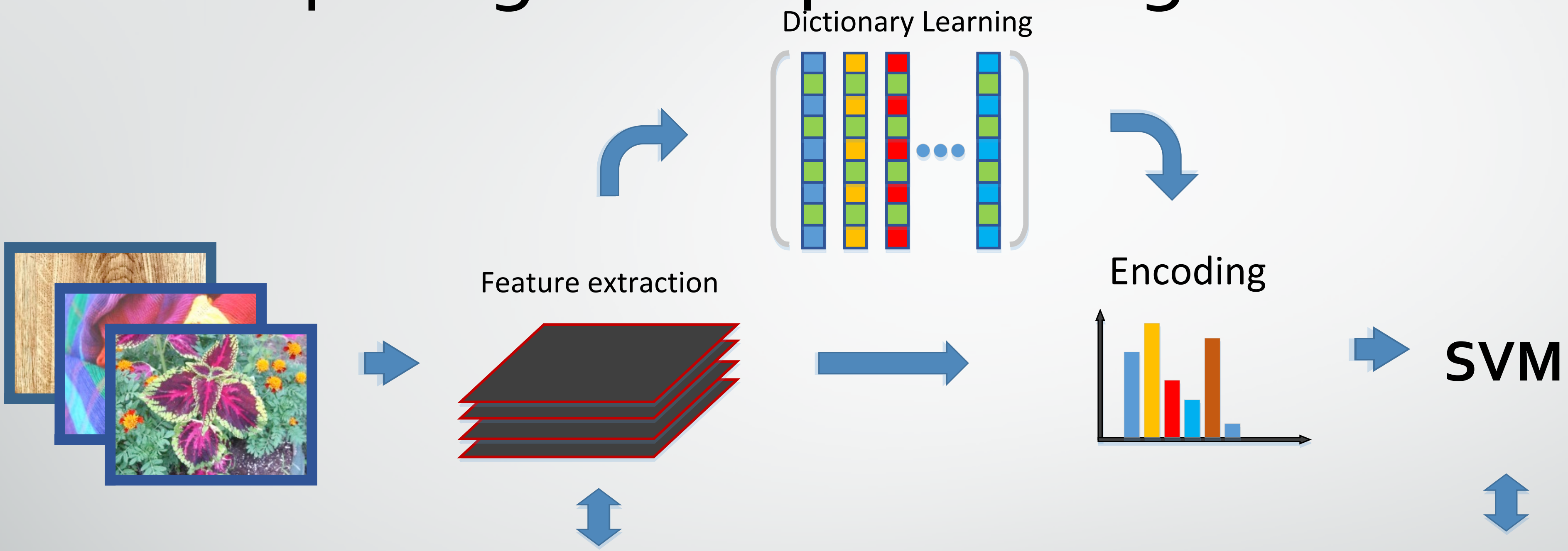


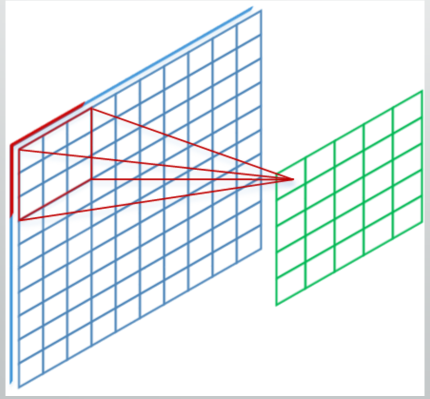
Classic Vision Approaches



- The input image sizes are flexible
- No domain-transfer problem

Comparing to Deep Learning Framework

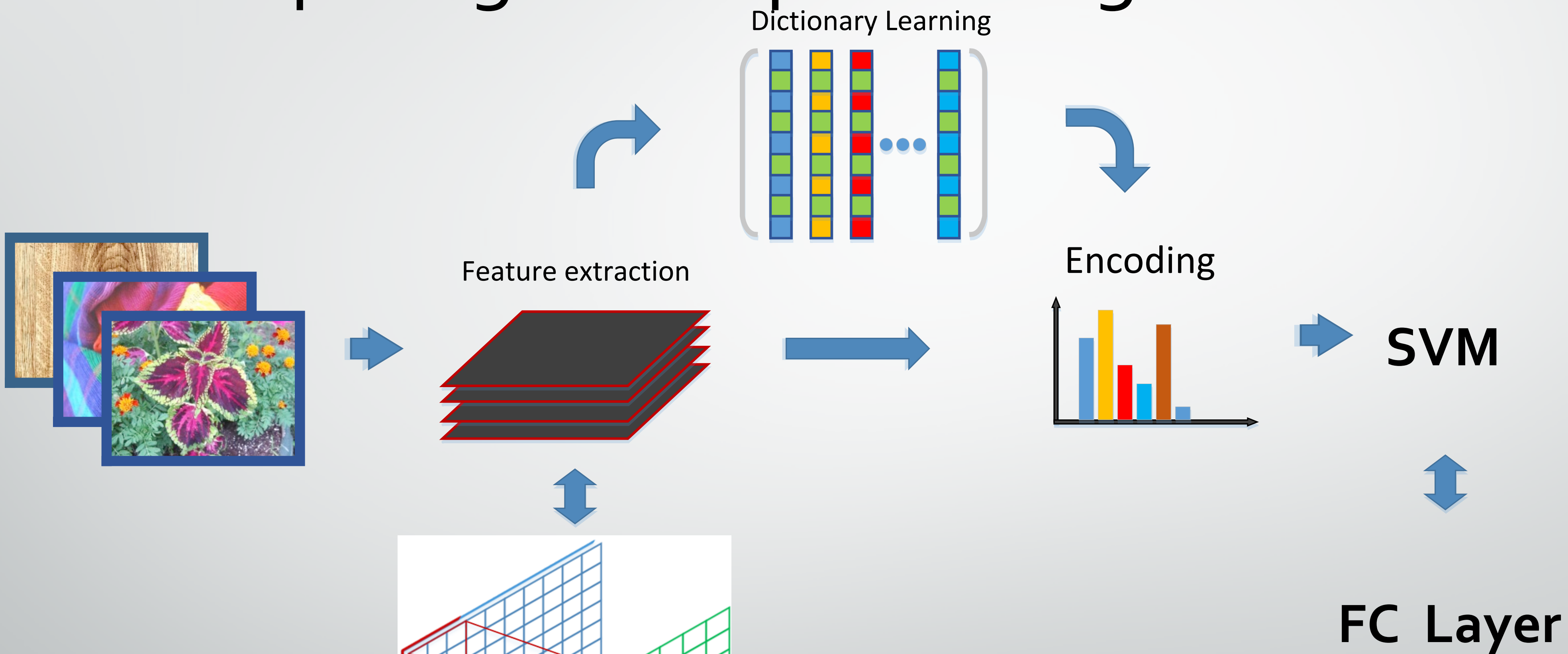


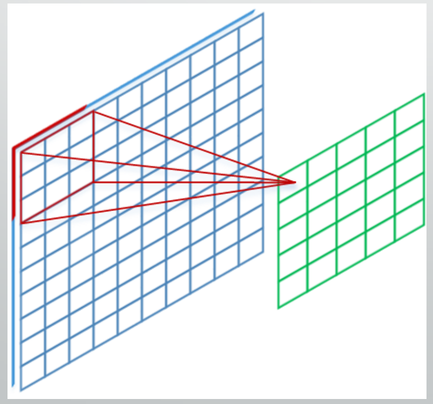


FC Layer

- Preserve Spatial Info
- Domain Transfer
- Fix size

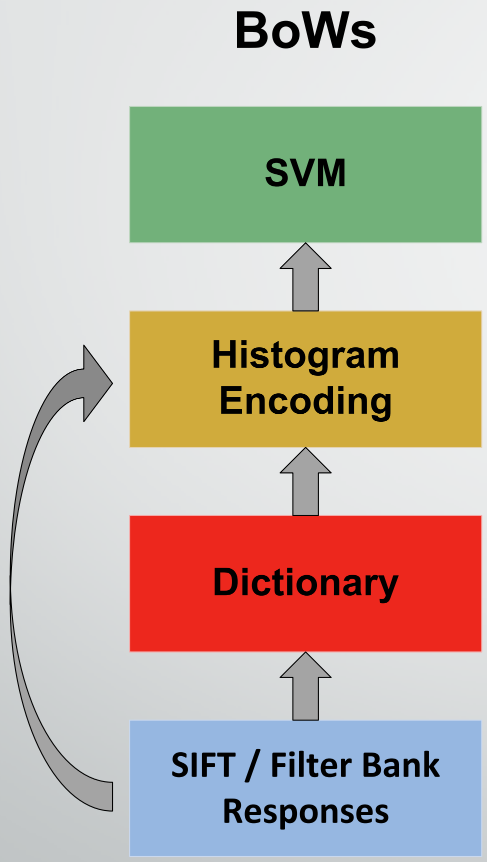
Comparing to Deep Learning Framework



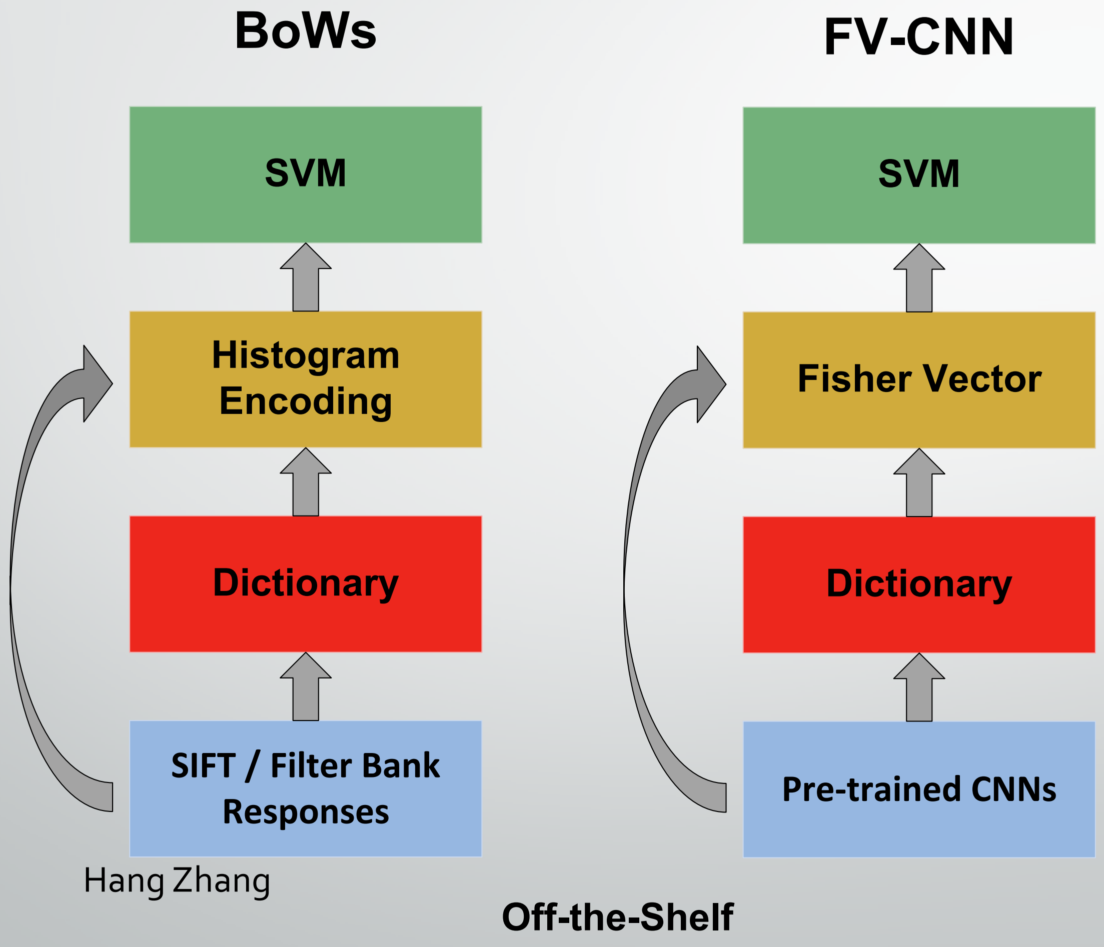


- Can we bridge the gap?

Hybrid Solution

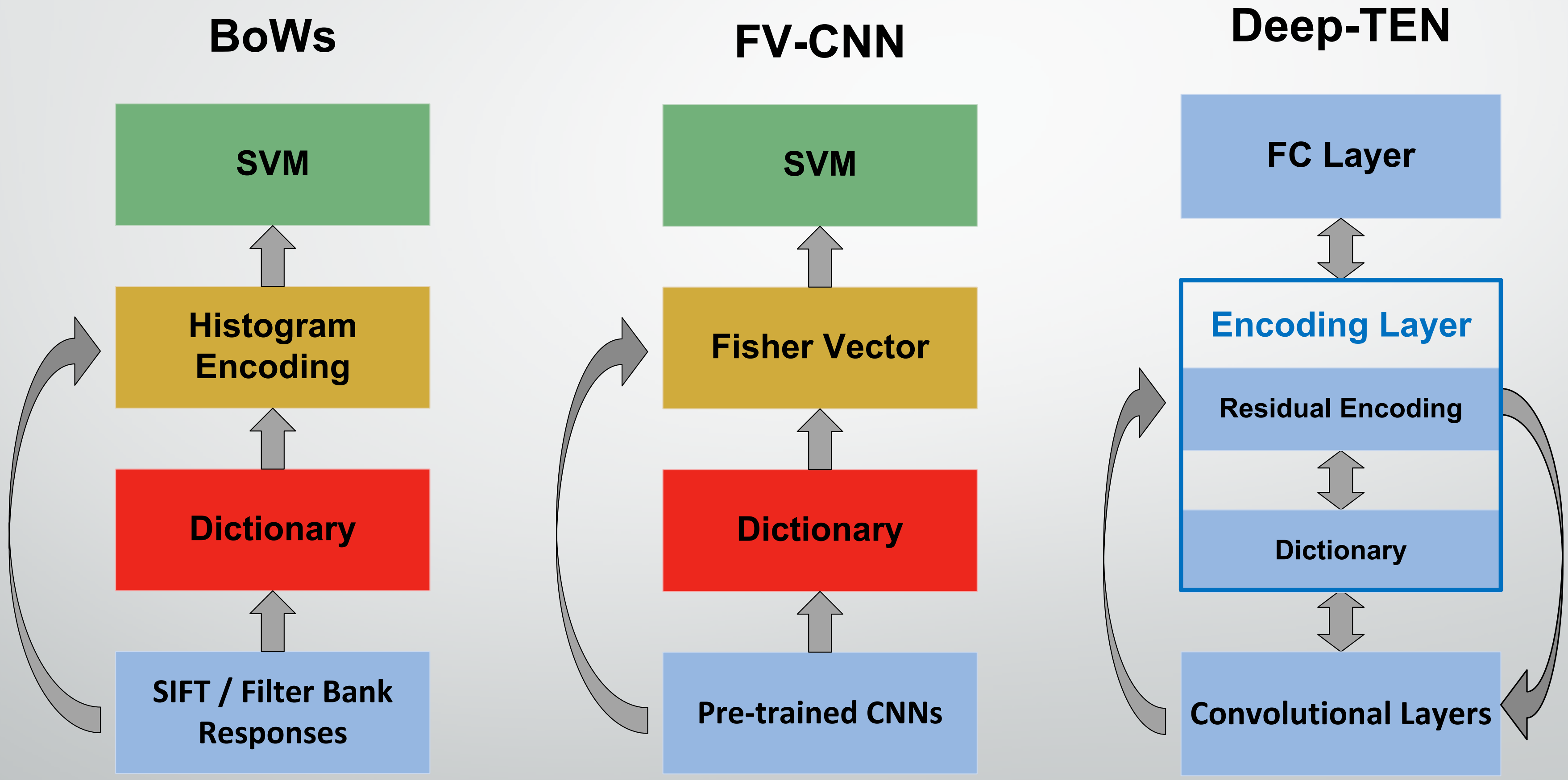


Hybrid Solution and Its Limitation



- Off-the-Shelf
- The dictionary and the encoders are fixed once built
- Feature learning and encoding are not benefiting from the labeled data

End-to-end Encoding



Off-the-Shelf

End-to-End

Bag-of-Words (BoW) Encoder

- Given a set of visual features $X = \{x_1, \dots x_N\}$, and a learned codebook $\mathrm{C} = \{c_1, \dots c_K\}$ (the input features is d-dimension and N is number of visual features and K is number of codewords)
- The assignment weight a_{ik} correspond to the visual feature x_i assigned to each codeword c_k. Hard-assignment: $a_{ik} = \delta(\|x_i - c_k\|^2 = \min_{j\in\{1,\dots K\}}\{\|x_i - c_j\|^2\})$
- BoWs counts the occurrences of the visual words $\sum_i a_i$

Residual Encoders

- The Fisher Vector, concatenating the gradient of GMM with respect to the mean and standard deviation

$$G_{d_k}^{X} = \sum_{i=1}^{N} a_{ik}\,(x_i - c_k)$$

$$G_{\sigma_k}^{X} = \sum_{i=1}^{N} a_{ik}\,[(x_i - c_k)^2 - 1]$$

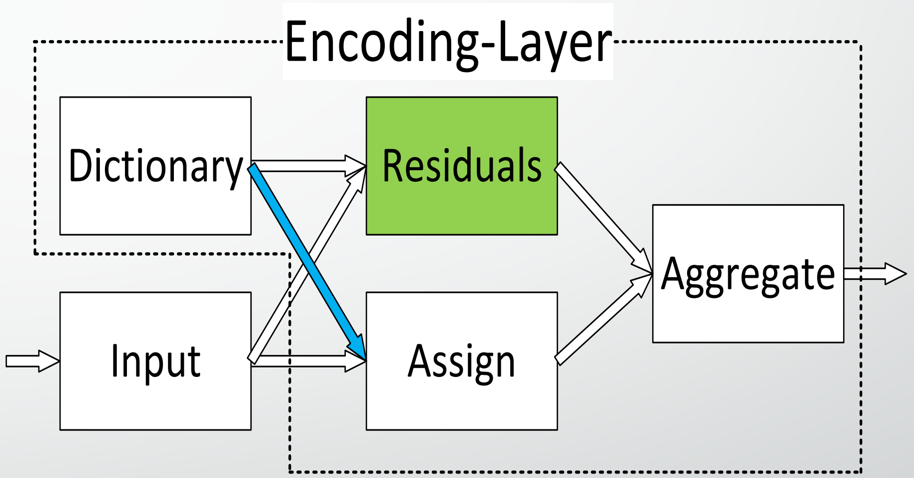
- VLAD (1st order, hard-assignment)

$$V_k = \sum_{i=NN(x_i)=d_k}^{N} (x_i - c_k)$$

Residual Encoding Model

- Residual vector $r_{ik} = x_i - c_k$
- Aggregating residuals with assignment weights

$$e_k = \sum_i a_{ik} r_{ik}$$



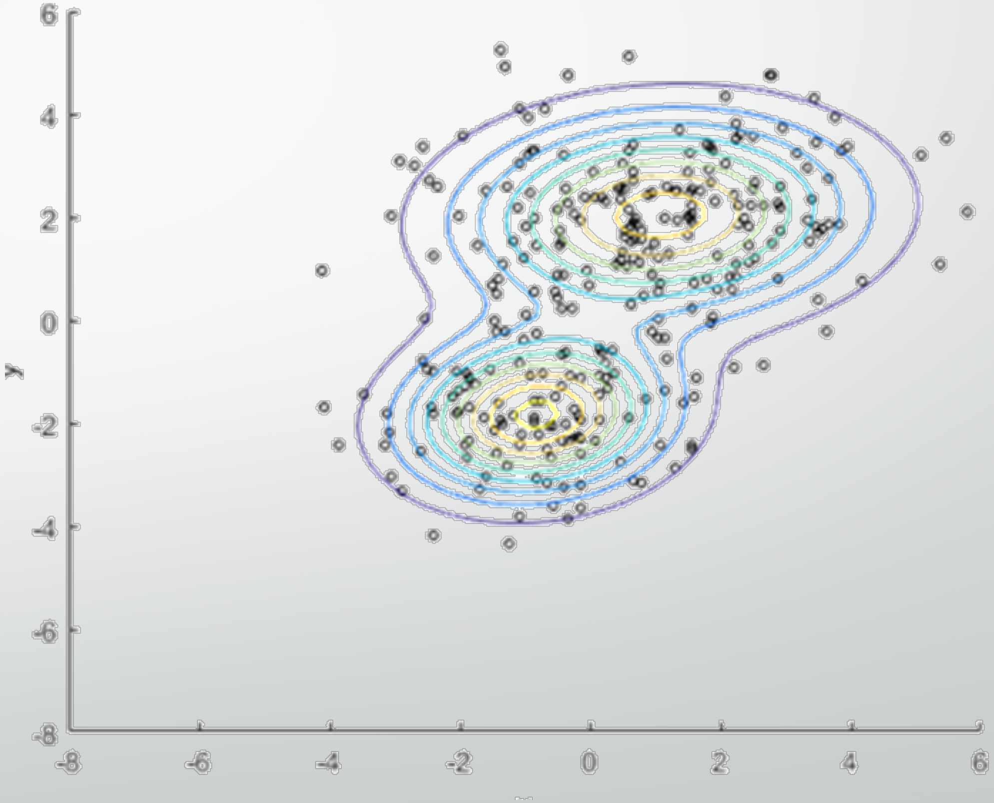
Feature Distributions and Assigning

- Soft-assignment

$$a_{ik} = \frac{\exp(-\beta \|r_{ij}\|^2)}{\sum_{j=1}^{K} \exp(-\beta \|r_{ij}\|^2)}$$

- Learnable Smoothing Factor

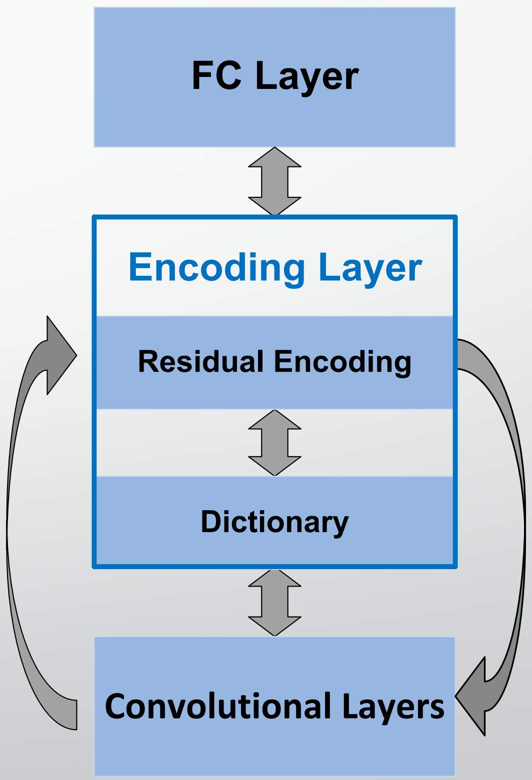
$$a_{ik} = \frac{\exp(-s_k \|r_{ik}\|^2)}{\sum_{j=1}^{K} \exp(-s_j \|r_{ij}\|^2)}$$



End-to-end Learning

- The loss function is differentiable *w.r.t* the input X and the parameters (Dictionary D and smoothing factors s)
- The Encoding Layer can be trained end-to-end by standard Stochastic Gradient Decent (SGD) with backpropagation

Deep-TEN



End-to-End

Gradients w.r.t Input X The encoder $E = \{e_1, ...e_K\}$ can be viewed as k independent sub-encoders. Therefore the gradients of the loss function ℓ *w.r.t* input descriptor x_i can be accumulated $\frac{d_\ell}{d_{x_i}} = \sum_{k=1}^{K} \frac{d_\ell}{d_{e_k}} \cdot \frac{d_{e_k}}{d_{x_i}}$. According to the chain rule, the gradients of the encoder $w.r.t$ the input is given by

$$\frac{d_{e_k}}{d_{x_i}} = r_{ik}^T \frac{d_{a_{ik}}}{d_{x_i}} + a_{ik} \frac{d_{r_{ik}}}{d_{x_i}}, \tag{4}$$

where a_{ik} and r_{ik} are defined in Sec 2, $\frac{d_{r_{ik}}}{d_{x_i}} = 1$. Let $f_{ik} = e^{-s_k \|r_{ik}\|^2}$ and $h_i = \sum_{m=1}^{K} f_{im}$, we can write $a_{ik} = \frac{f_{ik}}{h_i}$. The derivatives of the assigning weight *w.r.t* the input descriptor is

$$\frac{d_{a_{ik}}}{d_{x_i}} = \frac{1}{h_i} \cdot \frac{d_{f_{ik}}}{d_{x_i}} - \frac{f_{ik}}{(h_i)^2} \cdot \sum_{m=1}^{K} \frac{d_{f_{im}}}{d_{x_i}}, \tag{5}$$

where $\frac{d_{f_{ik}}}{d_{x_i}} = -2s_k f_{ik} \cdot r_{ik}$.

Gradients w.r.t Codewords C The sub-encoder e_k only depends on the codeword c_k. Therefore, the gradient of loss function *w.r.t* the codeword is given by $\frac{d_\ell}{d_{c_k}} = \frac{d_\ell}{d_{e_k}} \cdot \frac{d_{e_k}}{d_{c_k}}$.

$$\frac{d_{e_k}}{d_{c_k}} = \sum_{i=1}^{N} (r_{ik}^T \frac{d_{a_{ik}}}{d_{c_k}} + a_{ik} \frac{d_{r_{ik}}}{d_{c_k}}), \tag{6}$$

where $\frac{d_{r_{ik}}}{d_{c_k}} = -1$. Let $g_{ik} = \sum_{m \neq k} f_{im}$. According to the chain rule, the derivatives of assigning *w.r.t* the codewords can be written as

$$\frac{d_{a_{ik}}}{d_{c_k}} = \frac{d_{a_{ik}}}{d_{f_{ik}}} \cdot \frac{d_{f_{ik}}}{d_{c_k}} = \frac{2 s_k f_{ik} g_{ik}}{(h_i)^2} \cdot r_{ik}. \tag{7}$$

Gradients w.r.t Smoothing Factors Similar to the codewords, the sub-encoder e_k only depends on the k-th smoothing factor s_k. Then, the gradient of the loss function *w.r.t* the smoothing weight is given by $\frac{d_\ell}{d_{s_k}} = \frac{d_\ell}{d_{e_k}} \cdot \frac{d_{e_k}}{d_{s_k}}$.

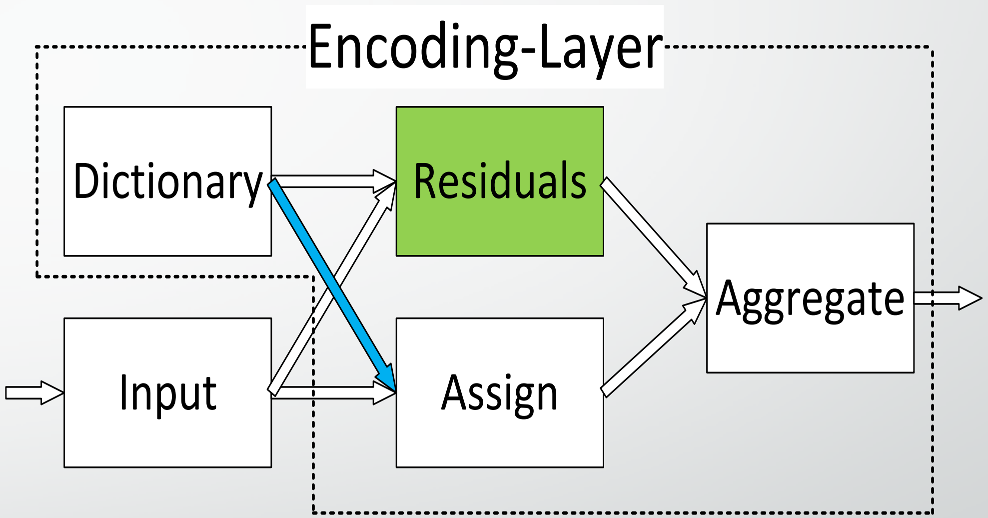
$$\frac{d_{e_k}}{d_{s_k}} = -\frac{f_{ik} g_{ik} \|r_{ik}\|^2}{(h_i)^2} \tag{8}$$

Relation to Dictionary Learning

- Dictionary learning approaches usually are achieved by **unsupervised** grouping (e.g. K-means) or minimizing the reconstruction error (e.g. K-SVD).
- The Encoding Layer makes the inherent dictionary differentiable *w.r.t* the loss function and learns the dictionary in a **supervised** manner.

Relation to BoWs and Residual Encoders

- Generalize BoWs, VLAD & Fisher Vector
- Arbitrary input sizes, output fixed length representation
- NetVLAD decouples the codewords with their assignments
  $a = f(x)$ instead of $a = f(x, d)$



Relation to Global Pooling Layer

- Sum Pooling (avg Pooling)
  Let $K = 1$ and $\mathrm{d} = 0$, then $e = \sum_{i=1}^{N} x_i$ and $\frac{d_l}{d_{x_i}} = \frac{d_l}{d_e}$
- SPP-Layer (He *et. al. ECCV 2014)*
  Fix bin numbers instead of receptive field, reshaping, arbitrary input size)
- Bilinear Pooling (Lin *et. al. ICCV 2015)*
  sum of the outer product across different location

Methods Overview

	Deep Features	Dictionary Learning	Residual Encoding	Any-size	Fine-tuning	End-to-end Classification
BoWs		✓		✓		
Fisher-SVM [40]		✓	✓	✓		
Encoder-CNN (FV [5] VLAD [18])	✓	✓	✓	✓		
CNN	✓				✓	✓
B-CNN [28]	✓				✓	
SPP-Net [19]	✓			✓	✓	✓
Deep TEN (ours)	✓	✓	✓	✓	✓	✓

Table 1: Methods Overview. Compared to existing methods, Deep-Ten has several desirable properties: it integrates deep features with dictionary learning and residual encoding and it allows any-size input, fine-tuning and provides end-to-end classification.

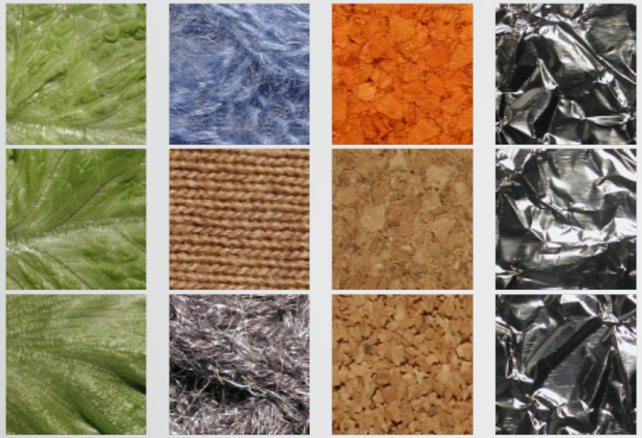
Domain Transfer

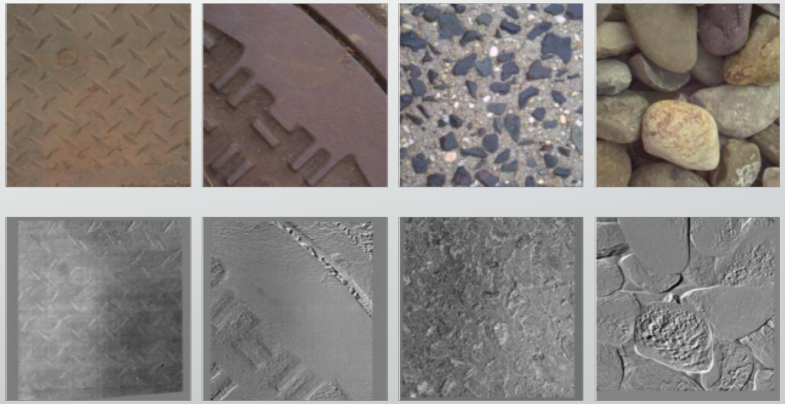
- The Residual Encoding Representation $e_k = \sum_i a_{ik} r_{ik}$
- For a visual feature x_i that appears frequently in the data
  - It is likely to close to a visual center d_k
  - e_k is close to zero, since $r_{ik} = x_i - d_k \approx 0$
  - e_j $(j \neq k)$ is close to zero, since $a_{ij} = \frac{\exp(-s_j \|r_{ij}\|^2)}{\sum_{m=1}^{K} \exp(-s_m \|r_{im}\|^2)} \approx 0$
- The Residual Encoding discard the frequently appearing features, which is like to be domain specific (useful for fine-tuning pre-trained features)

Experiments

- Datasets
  - Gold-standard material & texture datasets: *MINC-2500, KTH, FMD*
  - 2 Recent datasets: *GTOS, Light Field*
  - General recognition datasets: *MIT-Indoor, Caltech-101*
- Baseline approaches (off-the-shelf)
  - FV-SIFT (128 Gaussian Components, $32K \rightarrow 512$)
  - FV-CNN (Cimpoi *et. al.* pre-trained VGG-VD & ResNet, 32GMM)

Dataset Examples





(c) Plastic cover (d) Metal cover (e) Stone-cement (f) Pebble

Deep-TEN Architecture

	output size	Deep-TEN 50
Conv1	176×176×64	7×7, stride 2
Res1	88×88×256	3 × 3 max pool, stride 2 $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
Res2	44×44×512	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$
Res3	22×22×1024	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$
Res4	11×11×2048	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
Projection + Reshape	121×128	conv 1×1, 2048⇒128 W×H×D⇒N×D
Encoding	32×128	32 codewords
$L2$-norm + FC	n classes	1×1 FC

Table 2: Deep-TEN architectures for adopting 50 layer pre-trained ResNet. The 2nd column shows the featuremap sizes for input image size of 352×352. When multi-size training for input image size 320×320, the featuremap after Res4 is 10×10. We adopt a 1×1 convolutional layer after Res4 to reduce number of channels.

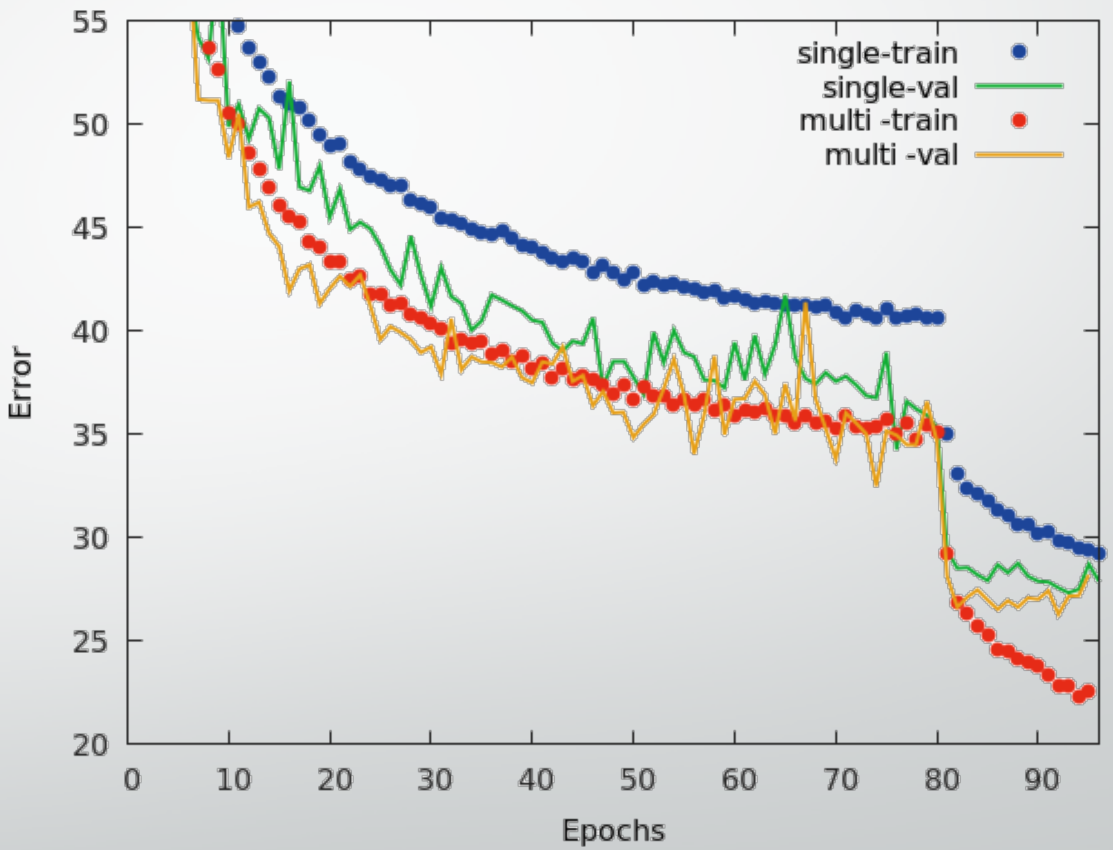
Comparing to the Baselines

	MINC-2500	FMD	GTOS	KTH	4D-Light	MIT-Indoor	Caltech-101
FV-SIFT	46.0	47.0	65.5	66.3	58.4	51.6	63.4
FV-CNN (VGG-VD)	61.8	75.0	77.1	71.0	70.4	67.8	83.0
Deep-TEN (ours)	**80.6**	**80.2**$\pm$0.9	**84.3**$\pm$1.9	**82.0**$\pm$3.3	**81.7**$\pm$1.0	**71.3**	**85.3**

Table 3: The table compares the recognition results of Deep-TEN with off-the-shelf encoding approaches, including Fisher Vector encoding of dense SIFT features (FV-SIFT) and pre-trained CNN activations (FV-CNN) on different datasets using single-size training. Top-1 test accuracy mean$\pm$std % is reported and the best result for each dataset is marked bold. (The results of Deep-TEN for FMD, GTOS, KTH datasets are based on 5-time statistics, and the results for MINC-2500, MIT-Indoor and Caltech-101 datasets are averaged over 2 runs. The baseline approaches are based on 1-time run.)

Multi-size Training (using different image sizes)

- Deep-TEN ideally accepts arbitrary sizes (larger than a constant)
- Training with predefined sizes iteratively in different epochs w/o modifying the solver
- Adopt single-size testing for simplicity



Multi-size Training

	MINC-2500	FMD	GTOS	KTH	4D-Light	MIT-Indoor
FV-CNN (VGG-VD) multi	63.1	74.0	79.2	77.8	76.5	67.0
FV-CNN (ResNet) multi	69.3	78.2	77.1	78.3	77.6	76.1
Deep-TEN (ours)	80.6	**80.2**$\pm$0.9	84.3$\pm$1.9	82.0$\pm$3.3	**81.7**$\pm$1.0	71.3
Deep-TEN (ours) multi	**81.3**	78.8$\pm$0.8	**84.5**$\pm$2.9	**84.5**$\pm$3.5	81.4$\pm$2.6	**76.2**

Table 4: Comparison of single-size and multi-size training.

Comparing to State-of-the-Art

	MINC-2500	FMD	GTOS	KTH	4D-Light
Deep-TEN* (ours)	**81.3**	80.2$\pm$0.9	**84.5**$\pm$2.9	**84.5**$\pm$3.5	**81.7**$\pm$1.0
State-of-the-Art	76.0$\pm$0.2 [2]	**82.4**$\pm$1.4 [5]	N/A	81.1$\pm$1.5 [4]	77.0$\pm$1.1 [43]

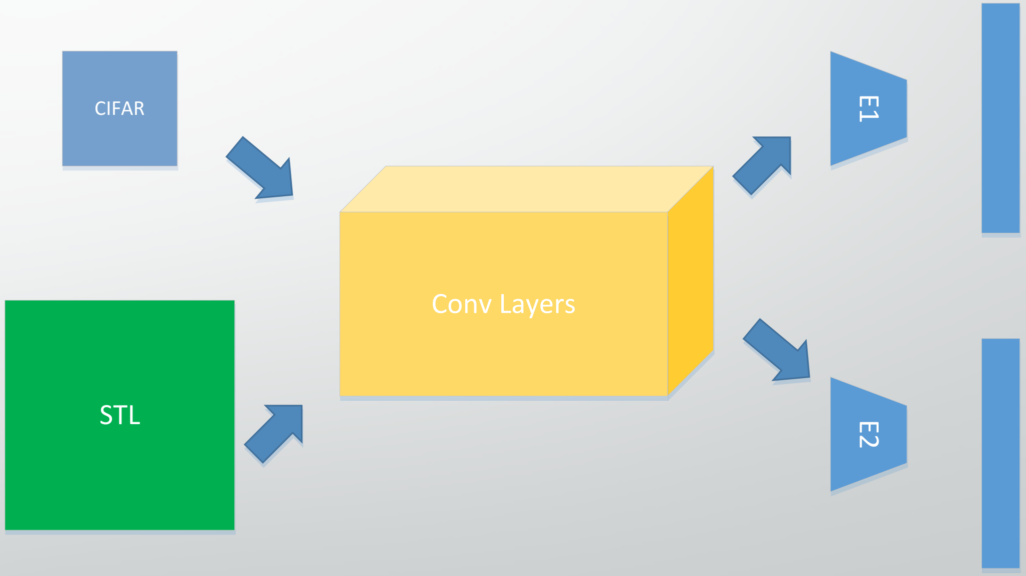
- Prior approaches
  - (1) relies on assembling features
  - (2)adopts an additional SVM classifier for classification.

Extra Thoughts

- So many labeled datasets: object recognition, scene understanding, material recognition
- How to benefit from them
  - Simply merging datasets (different label strategy)
  - Share convolutional features (domain transfer problem)

Joint Encoding

- Multi-task learning
- Encoding Layer carries the domain specific information
- Convolutional Layers are generic
- Joint training on two datasets
  - CIFAR-10 (50,000 training images with size 36×36)
  - STL-10 (5,000 training images with size 96×96)



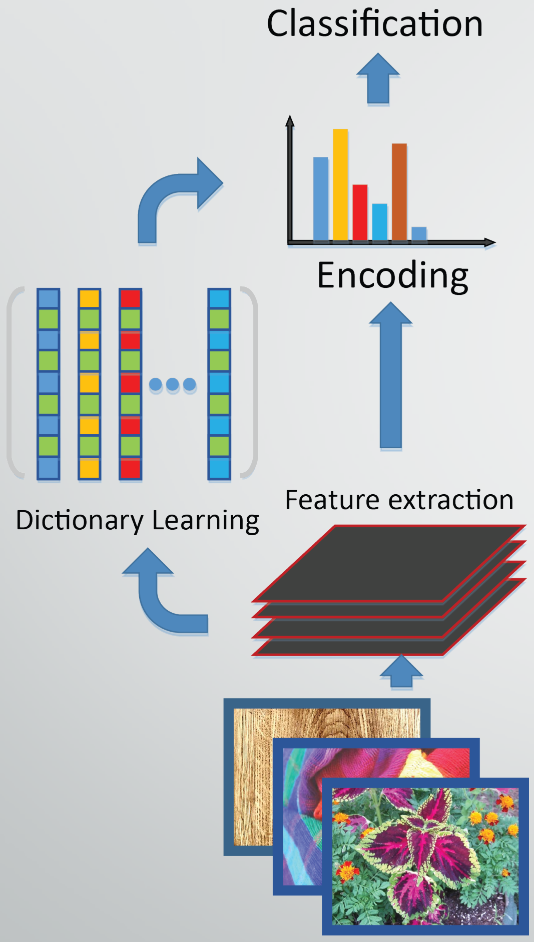
Experimental Results for Joint Training

- Joint training on two datasets (simple network architecture)
  - CIFAR-10 (50,000 training images with size 36×36)
  - STL-10 (5,000 training images with size 96×96)

	STL-10	CIFAR-10
Deep-TEN (Individual)	76.29	91.5
Deep-TEN (Joint)	**87.11**	91.8
State-of-the-Art	74.33 [49]	-

The SoA for CIFAR-10 is 95.4% using 1,001 layers ResNet (He *et. al. ECCV 2016*)

Summary



- Proposed a new model
  - Integrated the entire dictionary learning and encoding into a single layer of CNN
  - Generalize residual encoders (VLAD, FV), suitable for texture recognition and achieved state-of-the-art results
- Introduced a new CNN architecture
  - Making deep learning framework more flexible by allowing arbitrary input image sizes
  - Carries domain-specific information and make the learned features easier to transfer

Thank you!

- We provide efficient Torch implementation with CUDA backend at https://github.com/zhanghang1989/Deep-Encoding